

```
/*=====
Copyright July 2003 4G Color
All rights reserved
GD 7.18.03
=====*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define inputFileNAME "4gFilterForm"
#define outputFileNAME "4gParamArray"

typedef unsigned char U8 ;
typedef short UVAL; // uval is typically (-100,100)

enum colorDef { red, yellow, green, cyan, blue, magenta, all};
const int nColors = 7;
const int versionCode = 3314;

// filter basics =====
char * filterStrings[] = {
    "    red yellow green  cyan  blue magenta all          (+100,      -100)", // 0
    "    bright colors      (moreVivid, whiter)",           // 1
    "    deep colors        (moreVivid, blacker)",           // 2
    "    midtone colors     (moreVivid, grayer)",            // 3
    "    midtone lightness  (lighter,  darker)",            // 4
    "    whitePoint         (positive, negative)",           // 5
    "    lightGrayPoint     (positive, negative)",           // 6
    "    darkGrayPoint      (positive, negative)",           // 7
    "    blackPoint         (positive, negative)",           // 8
    "    color shift        (>rygcbm, <rygcbm)",             // 9
    "    light gray neutrals (whiter,  grayer)",             // 10
    "    dark gray neutrals  (blacker,  grayer)",            // 11
    "    textures           (sharper,  smoother)",          // 12
    "    edges              (sharper,  smoother)",          // 13
};

// filter structure, size is 2*((9*6)+4) = 116 bytes
typedef struct
{
    // neutral axis (NP) color correction
    UVAL  whiteNP      [nColors]; // (+,-)
    UVAL  lightGrayNP  [nColors]; // (+,-)
    UVAL  darkGrayNP   [nColors]; // (+,-)
    UVAL  blackNP      [nColors]; // (+,-)

    // neutral brightness correction
    UVAL  lightGray;    // (whiter,grayer)
    UVAL  darkGray;     // (blacker,grayer)

    // saturated color correction
    UVAL  colorShift    [nColors]; // (r>y>g>c>b>m>r,m<r<y<g<c<b<m)

    // tone correction
    UVAL  brightColors  [nColors]; // (moreVivid,whiter)
    UVAL  deepColors    [nColors]; // (moreVivid,blacker)
    UVAL  midToneColor  [nColors]; // (moreVivid,grayer)
    UVAL  midToneLightness[nColors]; // (lighter,darker)

    // acuity
    UVAL  texture;      // (sharper,smoother)
    UVAL  edges;        // (sharper,smoother)
} filterDEF;
```

```
void writeFilterForm(FILE *pForm, filterDEF fx)
{
    int i;
    fprintf( pForm, "%s\n",      filterStrings[0]);          // red yellow green  cyan  blue magent

    for(i=0; i<7; i++)
        fprintf( pForm, "%6d",  fx.brightColors[i] );
        fprintf( pForm, "%s\n", filterStrings[1]);          // (moreVivid,whiter)

    for(i=0; i<7; i++)
        fprintf( pForm, "%6d",  fx.deepColors[i] );
        fprintf( pForm, "%s\n", filterStrings[2]);          // (moreVivid,blackier)

    for(i=0; i<7; i++)
        fprintf( pForm, "%6d",  fx.midToneColor[i] );
        fprintf( pForm, "%s\n", filterStrings[3]);          // (moreVivid,grayer)

    for(i=0; i<7; i++)
        fprintf( pForm, "%6d",  fx.midToneLightness[i] );
        fprintf( pForm, "%s\n", filterStrings[4]);          // (lighter,darker)

    for(i=0; i<7; i++)
        fprintf( pForm, "%6d",  fx.whiteNP[i] );
        fprintf( pForm, "%s\n", filterStrings[5]);          // (+,-)

    for(i=0; i<7; i++)
        fprintf( pForm, "%6d",  fx.lightGrayNP[i] );
        fprintf( pForm, "%s\n", filterStrings[6]);          // (+,-)

    for(i=0; i<7; i++)
        fprintf( pForm, "%6d",  fx.darkGrayNP[i] );
        fprintf( pForm, "%s\n", filterStrings[7]);          // (+,-)

    for(i=0; i<7; i++)
        fprintf( pForm, "%6d",  fx.blackNP[i] );
        fprintf( pForm, "%s\n", filterStrings[8]);          // (+,-)

    for(i=0; i<7; i++)
        fprintf( pForm, "%6d",  fx.colorShift[i] );
        fprintf( pForm, "%s\n", filterStrings[9]);          // (lighter,darker)

    fprintf( pForm, "%42d%s\n", fx.lightGray, filterStrings[10]);
    fprintf( pForm, "%42d%s\n", fx.darkGray, filterStrings[11]);
    fprintf( pForm, "%42d%s\n", fx.texture, filterStrings[12]);
    fprintf( pForm, "%42d%s\n", fx.edges, filterStrings[13]);
}

// scan the user text form
void scanFilterForm(FILE *pForm, filterDEF *fx);
void scanFilterForm(FILE *pForm, filterDEF *fx)
{
    // we just scan uninteresting text with this buffer
    int i, temp;
    char buffer[100];
    fgets( buffer, 100, pForm );          // red yellow green  cyan  blue magent

    for(i=0; i<7; i++)
    {   fscanf( pForm, "%d", &temp);      fx-> brightColors[i] = temp;}
        fgets( buffer, 100, pForm );          // (moreVivid,whiter)

    for(i=0; i<7; i++)
    {   fscanf( pForm, "%d", &temp);      fx-> deepColors[i] = temp;}
        fgets( buffer, 100, pForm );          // (moreVivid,blackier)
```

```

for(i=0; i<7; i++)
{   fscanf( pForm, "%d", &temp);   fx-> midToneColor[i] = temp;}
    fgets( buffer, 100, pForm );           // (moreVivid,grayer)

for(i=0; i<7; i++)
{   fscanf( pForm, "%d", &temp);   fx-> midToneLightness[i] = temp;}
    fgets( buffer, 100, pForm );           // (lighter,darker)

for(i=0; i<7; i++)
{   fscanf( pForm, "%d", &temp);   fx-> whiteNP[i] = temp;}
    fgets( buffer, 100, pForm );           // (+,-)

for(i=0; i<7; i++)
{   fscanf( pForm, "%d", &temp);   fx-> lightGrayNP[i] = temp;}
    fgets( buffer, 100, pForm );           // (+,-)

for(i=0; i<7; i++)
{   fscanf( pForm, "%d", &temp);   fx-> darkGrayNP[i] = temp;}
    fgets( buffer, 100, pForm );           // (+,-)

for(i=0; i<7; i++)
{   fscanf( pForm, "%d", &temp);   fx-> blackNP[i] = temp;}
    fgets( buffer, 100, pForm );           // (+,-)

for(i=0; i<7; i++)
{   fscanf( pForm, "%d", &temp);   fx->colorShift[i] = temp;}
    fgets( buffer, 100, pForm );           // (lighter,darker)

fscanf( pForm, "%d", &temp);   fx-> lightGray = temp;   fgets( buffer, 100, pForm );
fscanf( pForm, "%d", &temp);   fx-> darkGray = temp;     fgets( buffer, 100, pForm );
fscanf( pForm, "%d", &temp);   fx-> texture = temp;       fgets( buffer, 100, pForm );
fscanf( pForm, "%d", &temp);   fx-> edges = temp;         fgets( buffer, 100, pForm );
}

// this is initialized sideways
void initFilter(filterDEF *fx);
void initFilter(filterDEF *fx)
{
    for(int i=0; i<nColors; i++)
    {
        fx-> whiteNP           [i] = 0;
        fx-> lightGrayNP       [i] = 0;
        fx-> darkGrayNP        [i] = 0;
        fx-> blackNP           [i] = 0;
        fx-> colorShift        [i] = 0;
        fx-> brightColors      [i] = 0;
        fx-> deepColors         [i] = 0;
        fx-> midToneColor       [i] = 0;
        fx-> midToneLightness   [i] = 0;
    }
    fx-> lightGray = 0;
    fx-> darkGray = 0;
    fx-> texture = 0;
    fx-> edges = 0;
}

void printFilter(filterDEF fp);
void printFilter(filterDEF fp)
{
    printf("The size of the filter is %5d bytes.\n", sizeof(fp)); //fa.lightGray);
    for(int i=0; i<nColors; i++)

```

```
{
    printf("%5d", fp.whiteNP[i]);
    printf("%5d", fp.lightGrayNP[i]);
    printf("%5d", fp.darkGrayNP[i]);
    printf("%5d", fp.blackNP[i]);
    printf("%5d", fp.colorShift[i]);
    printf("%5d", fp.brightColors[i]);
    printf("%5d", fp.deepColors[i]);
    printf("%5d", fp.midToneColor[i]);
    printf("%5d\n", fp.midToneLightness[i]);
}
printf("%5d\n", fp.lightGray);
printf("%5d\n", fp.darkGray);
printf("%5d\n", fp.texture);
printf("%5d\n", fp.edges);
}
// end filter basics =====

// Fog Filter =====
char * fogStrings[] = {
    "  TRANSFORM CREATOR (Compiles a compact image transform.)", // 0
    "Resize, Reshape, and Crop", // 1
    "  input pixels", // 2
    "  input lines", // 3
    "  output pixels", // 4
    "  output lines", // 5
    "  cropSelect      (inscribe, superscribe, anamorphic)", // 6
    "Exposure and Coding Correction", // 7
    "  auto Exposure    (autoExposureOff, autoExposure100, autoExposure80)", // 8
    "  auto Color Balance (autoColorBalanceOff, autoColorBalanceOn)", // 9
    "  jpeg Filter      (jpegOff, jpegOn)", // 10
    "Filter Selection and Composition", // 11
    "  filter select     (fA, fB, A+B, ABblend)", // 12
    "  filter A gain     (+,-)", // 13
    "  filter B gain     (+,-)", // 14
    "Foreground Filter--filter A", // 15
    "Background Filter--filter B", // 16
    "  Version code. Copyright 2003 by 4G Color. All rights reserved." // 17
};

typedef enum {inscribe, superscribe, anamorphic} cropDEF ;
typedef enum {fA, fB, AplusB, ABblend} compositionDEF;
typedef enum {autoExposureOff, autoExposure100, autoExposure80} autoExposureDEF;
typedef enum {autoColorBalanceOff, autoColorBalanceOn} autoColorBalanceDEF;
typedef enum {jpegOff, jpegOn} jpegFilterDEF;

// process structure
typedef struct
{
    // resize, reshape, and crop
    long    inPixels;
    long    inLines;
    long    outPixels;
    long    outLines;
    cropDEF cropSelect; // (inscribe, superscribe, anamorphic)

    // compensation
    autoExposureDEF    autoExposureSelect; // {autoExposureOff, autoExposure100, autoExposure80}
    autoColorBalanceDEF autoColorBalanceSelect; // {autoColorBalanceOff, autoColorBalanceOn}
    jpegFilterDEF    jpegFilterSelect; // {jpegOff, jpegOn}

    // filter composition and definition
    compositionDEF filterSelect; // {fA, fB, AplusB, ABblend}
    short          filterAGain; // (+,-)
```

```
    short      filterBgain;          // (+,-)
    filterDEF   filterA;
    filterDEF   filterB;
} fogParamArrayDEF;

void writeFogForm(FILE *pForm, fogParamArrayDEF fx)
{
    int temp = fx.filterAgain;
    fprintf( pForm, "%s\n",fogStrings[0]); // TRANSFORM CREATOR
    fprintf( pForm, "%s\n",fogStrings[1]); // resize, reshape, and crop",
    fprintf( pForm, "%6d%s\n", fx.inPixels, fogStrings[2]);
    fprintf( pForm, "%6d%s\n", fx.inLines, fogStrings[3]);
    fprintf( pForm, "%6d%s\n", fx.outPixels, fogStrings[4]);
    fprintf( pForm, "%6d%s\n", fx.outLines, fogStrings[5]);
    fprintf( pForm, "%6d%s\n", fx.cropSelect, fogStrings[6]);
    fprintf( pForm, "%s\n",fogStrings[7]); // compensation
    fprintf( pForm, "%6d%s\n", fx.autoExposureSelect, fogStrings[8]);
    fprintf( pForm, "%6d%s\n", fx.autoColorBalanceSelect, fogStrings[9]);
    fprintf( pForm, "%6d%s\n", fx.jpegFilterSelect, fogStrings[10]);
    fprintf( pForm, "%s\n",fogStrings[11]); // filter composition and definition
    fprintf( pForm, "%6d%s\n", fx.filterSelect, fogStrings[12]);
    fprintf( pForm, "%6d%s\n", fx.filterAgain, fogStrings[13]);
    fprintf( pForm, "%6d%s\n", fx.filterBgain, fogStrings[14]);

    fprintf( pForm, "%s\n",fogStrings[15]); // filter A
    writeFilterForm(pForm, fx.filterA);

    fprintf( pForm, "%s\n",fogStrings[16]); // filter B
    writeFilterForm(pForm, fx.filterB);

    fprintf( pForm, "%6d%s\n", versionCode, fogStrings[17]);
}

// scan the fogForm
int scanFogForm(FILE *pForm, fogParamArrayDEF *fx);
int scanFogForm(FILE *pForm, fogParamArrayDEF *fx)
{
    int temp;
    char buffer[100];

    fgets( buffer, 100, pForm ); // TRANSFORM CREATOR
    fgets( buffer, 100, pForm ); // resize, reshape, and crop",
    fscanf( pForm, "%d", &temp); fx-> inPixels = temp; fgets( buffer, 100, pForm );
    fscanf( pForm, "%d", &temp); fx-> inLines = temp; fgets( buffer, 100, pForm );
    fscanf( pForm, "%d", &temp); fx-> outPixels = temp; fgets( buffer, 100, pForm );
    fscanf( pForm, "%d", &temp); fx-> outLines = temp; fgets( buffer, 100, pForm );
    fscanf( pForm, "%d", &temp); fx-> cropSelect = (cropDEF)temp; fgets( buffer, 100, pF
    fgets( buffer, 100, pForm ); // compensation
    fscanf( pForm, "%d", &temp); fx-> autoExposureSelect = (autoExposureDEF)temp; fgets(
    fscanf( pForm, "%d", &temp); fx-> autoColorBalanceSelect = (autoColorBalanceDEF)temp;
    fscanf( pForm, "%d", &temp); fx-> jpegFilterSelect = (jpegFilterDEF)temp; fgets( buf
    fgets( buffer, 100, pForm ); // filter composition and definition
    fscanf( pForm, "%d", &temp); fx-> filterSelect = (compositionDEF)temp; fgets( buffer,
    fscanf( pForm, "%d", &temp); fx-> filterAgain = temp; fgets( buffer, 100, pForm );
    fscanf( pForm, "%d", &temp); fx-> filterBgain = temp; fgets( buffer, 100, pForm );

    fgets( buffer, 100, pForm ); // filter A
    scanFilterForm(pForm, &fx->filterA);

    fgets( buffer, 100, pForm ); // filter B
    scanFilterForm(pForm, &fx->filterB);

    fscanf( pForm, "%d", &temp, fogStrings[17]);
    return temp;
```

```
}
```

```
void initFog(fogParamArrayDEF *pax);
void initFog(fogParamArrayDEF *pax)
{
    // resize, reshape, and crop
    pax-> inPixels      = 0;
    pax-> inLines       = 0;
    pax-> outPixels     = 0;
    pax-> outLines      = 0;
    pax-> cropSelect    = inscribe;

    // compensation
    pax-> autoExposureSelect    = autoExposureOff;
    pax-> autoColorBalanceSelect = autoColorBalanceOff;
    pax-> jpegFilterSelect      = jpegOff;

    // filter composition and definition
    pax-> filterSelect = fA;
    pax-> filterAgain  = 100;
    pax-> filterBgain  = 100;

    initFilter( &(amp;pax->filterA) );
    initFilter( &(amp;pax->filterB) );
}
```

```
void printFog(fogParamArrayDEF pa);
void printFog(fogParamArrayDEF pa)
{
    printf("%5d\\n", pa.inPixels);
    printf("%5d\\n", pa.inLines);
    printf("%5d\\n", pa.outPixels);
    printf("%5d\\n", pa.outLines);
    printf("%5d\\n", pa.cropSelect);

    printf("%5d\\n", pa.autoExposureSelect);
    printf("%5d\\n", pa.autoColorBalanceSelect);
    printf("%5d\\n", pa.jpegFilterSelect);

    printf("%5d\\n", pa.filterSelect);
    printf("%5d\\n", pa.filterAgain);
    printf("%5d\\n", pa.filterBgain);

    printFilter( pa.filterA);
    printFilter( pa.filterB);
}
```

```
// End Fog Filter =====
```

```
void clip( filterDEF *pf)
```

```
{
    int nn = nColors-1;

    // merge values
    for(int i=0; i<nn; i++ )
    {
        // neutral axis (NP) color correction
        pf-> whiteNP      [i] += pf-> whiteNP      [nn];
        pf-> lightGrayNP  [i] += pf-> lightGrayNP  [nn];
        pf-> darkGrayNP   [i] += pf-> darkGrayNP   [nn];
        pf-> blackNP      [i] += pf-> blackNP      [nn];
        // saturated color correction
        pf-> colorShift   [i] += pf-> colorShift   [nn];
        // tone correction
        pf-> brightColors [i] += pf-> brightColors [nn];
    }
}
```

```
        pf-> deepColors      [i] += pf-> deepColors      [nn];
        pf-> midToneColor    [i] += pf-> midToneColor    [nn];
        pf-> midToneLightness [i] += pf-> midToneLightness [nn];
    }

    int count = sizeof(*pf)/sizeof(UVAL); // size of the parameter array
    UVAL *aa = (UVAL *)pf; // directly address the parameter array

    // clip values to +-100 and rerange to +-255
    for(int i=0; i<count; i++ )
    {
        aa[i] = (255*aa[i])/100;
        if( aa[i]>255 ) aa[i]=255;
        if( aa[i]<-255 ) aa[i]=-255;
    }
}

// clip, merge and rerange parameters
//void image::translate()
//{
//    clip( &pp.filterA );
//    clip( &pp.filterB );
//}

int main()
{
    fogParamArrayDEF xxx;
    FILE * pForm;

    pForm = fopen (outputFILENAME, "rb");
    if( pForm==NULL ) goto PAUSE;
    fread( &xxx, sizeof(xxx), 1, pForm );
    fclose(pForm);

    clip( &xxx.filterA );
    clip( &xxx.filterB );

    pForm = fopen (outputFILENAME, "wb");
    if( pForm==NULL ) goto PAUSE;
    fwrite( &xxx, sizeof(xxx), 1, pForm );
    fclose(pForm);

    return 0;
PAUSE:
    printf("\n      Error. (Enter a number to exit.)\n");
    int temp;
    scanf("%d", &temp);
    printf("\n      ... bye\n");
}

// copyright 2003, 4G Color, All rights reserved
```